Задания к работе №3 по фундаментальным алгоритмам (2022-2023 уч. г.).

Приложения не должны завершаться аварийно.
Во всех заданиях запрещено пользоваться функциями exit(int), perror(const char*) и т. п. (позволяющими завершить выполнение приложения из произвольной точки выполнения).
Во всех заданиях при реализации необходимо разделять контексты работы с данными (поиск, сортировка, добавление/удаление, модификация и т. п.) и вывод данных в поток вывода.
Во всех заданиях все вводимые (с консоли, файла, командной строки) пользователем данные должны подвергаться валидации в соответствии с типом валидируемых данных.
Во всех заданиях необходимо контролировать ситуации с невозможностью [пере]выделения памяти.

Дополнительные требования для заданий (см. в тексте задания):
[1] - Операция перевыделения памяти для Вашей реализации динамического массива должна выполняться по времени за амортизированную константу.
[2] - В процессе работы приложения и перед выходом из него необходимо корректно очистить всю выделенную динамическую память.
[3] - (при чтении файла проверяйте валидность считываемых данных (количество значений в строке и сами значения), если данные невалидны - пропускайте текущую строку файла)
[4] - (при чтении файла проверяйте валидность считываемых данных (количество значений в строке и сами значения), если данные невалидны - верните в вызывающий код соответствующий результат, а в самом вызывающем коде - обработайте этот результат)
[5] - (которые необходимо проверить на корректность, в случае некорректного ввода - необходимо вывести на консоль предупреждение и запросить повторный ввод данных; после трёх попыток некорректного ввода необходимо завершить выполнение приложения путём возврата из функций в стеке вызовов результатов работы и их обработки в вызывающем коде)
[6] - (для сортировки: лексикографический компаратор)
[7] - Все ошибки, связанные с операциями открытия файла, должны быть обработаны; все открытые файлы должны быть закрыты.
[8] - (строка в формате “{день}.{месяц}.{год} {часы}:{минуты}:{секунды}”, являющаяся корректным представлением даты и времени по григорианскому календарю; год >= 2000; часы в диапазоне [0..23], минуты в диапазоне [0..59], секунды в диапазоне [0..59])
[9] - (строка в формате “{день}.{месяц}.{год}”, являющаяся корректным представлением даты и времени по григорианскому календарю; год >= 2000)

1. Реализуйте функцию перевода целого числа из десятичной системы счисления в систему счисления с основанием $2^r,\ r = 1, ..., 5$ (параметрами функции являются переводимое число и значение r - степень двойки; параметры необходимо проверить на корректность). При реализации функции разрешается использовать битовые операции и операции обращения к памяти, запрещается использовать стандартные арифметические операции. Продемонстрируйте работу реализованной функции.

2. При реализации функций из данного задания запрещается пользоваться строковым представлением числа в двоичной системе счисления. Параметры, фигурирующие в условии, необходимо считать с консоли (и проверить их на корректность). Обе функции:
   - возвращают динамический массив найденных чисел, от меньшего к большему, через свой параметр;
   - возвращают длину массива через свой параметр;
   - имеют тип возвращаемого значения void.

1. Реализовать функцию поиска всех $k-$ битных целых чисел типа int, в двоичной записи которых присутствует ровно $l,\ l \leq k$ единиц.
2. Реализовать функцию поиска всех $k-$ битных целых чисел типа int, в двоичной записи которых присутствует ровно $l,\ l \leq k$ подряд идущих единиц.
Продемонстрируйте работу реализованных функций.

3. На вход программе через аргументы командной строки подается файл и флаг (флаг начинается с символа ‘-’ или ‘/’, второй символ - ‘a’ или ‘d’). В файле в каждой строке содержится информация о сотруднике (поля структуры типа employee): id (неотрицательное целое число), имя (непустая строка только из символов букв латинского алфавита), фамилия (непустая строка только из символов букв латинского алфавита), заработная плата (неотрицательное вещественное число). Необходимо реализовать функцию, считывающую записи из файла [3] и возвращающую динамический массив структур типа employee. В функции main необходимо получить массив структур из файла, путь к которому передаётся через аргументы командной строки. Далее необходимо записать в выходной файл (путь - аргумент командной строки), считанный массив, отсортированный (с аргументом командной строки: “-a” - по возрастанию, с флагом “d” - по убыванию) по зарплате. Сортировку коллекции структур реализуйте на базе структуры данных типа фибоначчиева пирамида, которую также реализуйте самостоятельно. [1] [2] [7]

4. Структура message содержит следующие поля: id сообщения (натуральное число), текст сообщения (строка), длина сообщения в байтах (целое неотрицательное число). Приложение предлагает пользователю ввести сообщение, после чего относительно введённых данных [5] происходит заполнение переменной (объекта) структуры и запись данных этого объекта в файл со структурой CSV (имя файла должно генерироваться псевдослучайно: в имени файла каждый символ равновероятно может являться символом буквы или символом цифры; символы букв равновероятны, символы цифр равновероятны; расширение файла - ".csv"). Когда пользователь вводит сообщение, текст которого передаётся приложению как аргумент командной строки (непустая строка), ввод данных завершается, и приложение должно считать данные из записанного файла в динамический массив структур типа message и вывести их на консоль; вывод на консоль данных из переменной структуры типа message организуйте при помощи своей функции. [1] [2] [7]

5. Структура student содержит следующие поля: id студента (натуральное число), имя (непустая строка только из символов букв латинского алфавита), фамилия (непустая строка только из символов букв латинского алфавита), курс (натуральное число в диапазоне [1..4]), группа (непустая строка) и оценки за 5 экзаменов (динамический массив целых чисел в диапазоне [2..5]). Через аргументы командной строки приложению подается путь к файлу, содержащему информацию о студентах. Приложение должно считать из файла информацию о студентах в динамический массив объектов структур [4]. В приложении должны быть реализованы функции поиска (с возвратом коллекции найденных значений) и сортировка (используйте функцию qsort) студента(-ов) по id, по фамилии [6], по имени [6], по группе [6]; а также функция группировки студентов по курсу (возвращаемое значение - массив списков структур student). Реализуйте функцию вывода в поток данных из объекта структуры student: ФИО студента, курс, группу и среднюю оценку за экзамены. Далее необходимо в трассировочные файлы (для каждого курса - свой файл; формат пути к файлу: "{аргумент командной строки}_{номер курса}") вывести информацию о студентах, чей средний балл за все экзамены выше среднего. [1] [2] [7]

6. 1. Реализуйте структуру string, содержащую в себе поля для динамического массива символов (указатель на char) и длины строки (целое неотрицательное число). Для структуры реализуйте функционал: печати данных (массива символов) в переданный поток, создания объекта строки, удаления данных из объекта строки, сравнения двух объектов строк по переданному компаратору, копирования данных из одного объекта строки в другой, конкатенации произвольного числа строк (с переменным числом аргументов), дублирования строки (создания копии объекта строки на основе переданного в функцию). Продемонстрировать работу функционала для структуры string. [2]
2. Структура mail содержит объекты структуры address (поля структуры: город (непустая строка только из символов букв латинского алфавита), улица (непустая строка только из символов букв латинского алфавита), дом (натуральное число), квартира (натуральное число), индекс (строка из шести символов цифр)), для отправителя и получателя, вес посылки (вещественное неотрицательное число), почтовый идентификатор (строка из четырнадцати символов цифр), время создания [8], время вручения [8]. Структура post содержит указатель на переменную структуры address текущего почтового отделения и динамический массив структур типа mail. Реализуйте в интерактивном диалоге с пользователем добавление и удаление объектов структур типа mail в переменной (объекте) структуры типа post, информативный вывод при поиске по почтовому идентификатору. Объекты структур типа mail в объектах структур типа post должны быть отсортированы по индексу получателя (первично) и почтовому идентификатору посылки (вторично) в произвольный момент времени. Добавьте в диалог опции поиска всех доставленных на текущий момент времени (аргумент командной строки [8]) отправлений, а также всех отправлений, срок доставки которых истек. Коллекции найденных доставленных/недоставленных отправлений должны быть отсортированы по дате отправления (первично) и почтовому идентификатору (вторично). Для хранения строковых данных используйте структуру string и функционал для этой структуры из п. 1. [2]

7. В текстовом файле (путь к файлу передаётся как аргумент командной строки) находится информация о жителях некоторого поселения: фамилия (непустая строка только из символов букв латинского алфавита), имя (непустая строка только из символов букв латинского алфавита), отчество (непустая строка только из символов букв латинского алфавита), дата рождения [9], пол (определите свой тип перечисления с двумя именованными константами), средний доход за месяц (неотрицательное вещественное число). Реализуйте приложение, считывающее эту информацию из файла в односвязный упорядоченный список структур типа (в порядке увеличения возраста). Реализуйте возможности поиска жителя с заданными параметрами, удаления/добавления информации о жителях и возможность выгрузки данных из списка обратно в файл. Продемонстрируйте работу с реализованным функционалом и обеспечьте наглядный вывод результатов работы функций. [7]

8. 1. Реализуйте функцию для сбора статистических данных по заданному тексту. Результатом работы функции должна являться сформированная в виде объекта структуры бинарного дерева поиска информация о том, сколько раз каждое слово (лексема) из файла встречается в данном файле. Реализуйте опции: определить, сколько раз заданное слово встретилось в файле; возврата коллекции первых n (параметр функции) наиболее часто встречающихся слов в файле; поиска самого большого и самого маленького слова по компаратору: первичное сравнение длин строк, вторичное - лексикографическое сравнение строк. В процессе построения результирующего дерева, для определения наличия или отсутствия в промежуточном состоянии объекта бинарного дерева очередной обрабатываемой лексемы, в процессе построения результирующего дерева поддерживайте сопряжённо с результирующим деревом вспомогательное префиксное дерево из символов лексем (каждый элемент префиксного дерева должен хранить динамический массив указателей на поддеревья, по которому возможно выполнение алгоритма дихотомического поиска; значение текущего символа лексемы; указатель на элемент из результирующего дерева). Файл должен обрабатываться при помощи метода границы, для которого нужно иметь возможность конфигурировать функцию отделения “хороших” (входящих в лексемы) символов от “плохих” (разделителей лексем);
2. Для построенного дерева из п. 1. реализуйте рекурсивную функцию поиска глубины данного дерева;
3. Реализуйте функции сохранения и восстановления бинарного дерева в/из файл(а). При этом восстановленное из файла дерево должно иметь точно такую же структуру и вид, как и до сохранения.
Продемонстрируйте работу функций (поочерёдно п.1, п.2, п.3) на примере файла, путь к которому является аргументом командной строки. Организуйте наглядный вывод результатов. [1] [2]

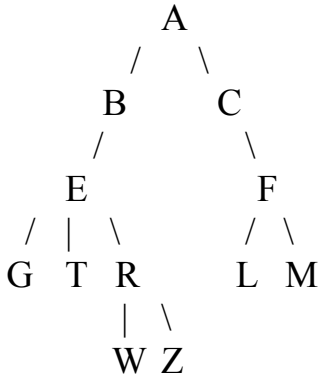
9. 1. Реализуйте приложение, моделирующее приоритетную очередь сообщений на основе двусвязного списка. Для приоритетной очереди реализуйте операции вставки элемента, возврата минимума (максимума) и извлечения минимума (максимума). Данными, которые хранятся в очереди, являются объекты структур типа T с полями текст (строка из п. 6.1) и приоритет (целое неотрицательное число в диапазоне [0..10]). При вставке нового элемента в очередь, его позиция в очереди определяется на основе его приоритета: чем больше значение приоритета (по компаратору для int по умолчанию), тем ближе к началу очереди он добавляется, причём элементы с одинаковым приоритетом должны находиться в очереди в порядке их вставки. Входными данными (аргументами) приложения являются пути к текстовым файлам, имеющим следующий формат:

prior=0 task='Задание 1'
prior=0 task='Задание 3'
prior=5 task='Задание 2'
*prior=2 task='***Где лабы***'*
prior=3 task='Задание 3'
prior=3 task='Задание 3'
prior=2 task='Задание 5'

Количество путей к входным файлам не ограничено и может быть произвольным. Поочерёдно из каждого файла (пояснение: берём первую строку первого файла, затем первую строку второго файла, …, первую строку последнего файла, вторую строку первого файла, …, последнюю строку последнего файла) необходимо считывать строки, вычитывать данные и сохранять их в объекты структур типа T с последующим добавлением в приоритетную очередь. По завершении обработки всех файлов необходимо освободить очередь путём удаления всех элементов через операцию extract_min с печатью значений полей приоритета и текста извлечённого элемента в выходной файл. [2] [7]

2. Для задания 9.1 замените приоритетную очередь на базе двусвязного списка на приоритетную очередь на базе двоичной кучи. Сравните эффективность (по времени) операций реализованных структур, а также время работы Ваших реализаций. [2] [7]

10. 1. Реализуйте приложение для построения дерева скобочного выражения. На вход программе подается файл, в котором содержится произвольное число строк; каждая строка является корректным скобочным выражением. Приложение должна обработать каждое скобочное выражение из файла и вывести в текстовый файл деревья скобочных записей в наглядной форме (форму вывода определите самостоятельно). Возникающие при работе программы деревья не обязаны быть бинарными. Например, запись A(B(E(G,T,R(W,Z))),C(F(L,M))) соответствует дереву



Формат вывода не фиксирован и определяется удобством восприятия.

2. Пользователь вводит с клавиатуры различные инструкции, которые являются строками. При этом каждые N (аргумент командной строки) строк, которые ввел пользователь, сохраняются в текстовый файл и у пользователя есть возможность удалить последние $N/2$ введенных строк. Организовать хранение строк в программе в виде односвязного списка, при этом, сохранение строк реализовать посредством очереди, которая содержит введенные команды, а возможность отмены, реализовать с помощью стека, который содержит те же самые элементы. [2]